TABLE NO

STUDENT ID NO

MULTIMEDIA UNIVERSITY®

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2019/2020

THP7021 – HIGH PERFORMANCE COMPUTING AND BIG DATA

(All sections / Groups)

27 September 2019
2.30 p.m - 4.30 p.m
(2 Hours)

Q1	
Q2	
Q3	
Q4	
Total	

INSTRUCTIONS TO STUDENT

1. This Question paper consists of 11 pages with 4 Questions only.
2. Attempt **ALL FOUR (4)** questions. All questions carry equal marks and the distribution of the marks for each question is given.
3. Please print all your answers in this question paper.

Question 1 [25 marks]

This serial program in *Listing 1*, if asked to calculate a picture of size 1024, will produce a picture like what is shown in *Figure 1*.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(int argc, char*argv[])
{ int xmax, ymax, x, y, size;
  double divisor;
  int **picture;
  char buff[1024];
  FILE *outfile;

  printf("What file should I store the picture in? ");fflush(stdout);
  gets(buff);
  printf("How big should the picture be? ");fflush(stdout);
  scanf("%d", &size);

  divisor = (M_PI*size/60);
  picture =(int**)malloc(sizeof(int*)*size);

  for (y=0; y<size; y++)
  { picture[y] = (int*)malloc(sizeof(int)*size);
    for (x=0; x<size; x++)
      picture[y][x] = (sin(x/divisor)+cos(y/divisor))*64+128;
  }


  outfile=fopen(buff,"w");
  fprintf(outfile,"P2\n%d %d 256\n",size, size);
  for(y=0; y<size; y++)
    for (x=0; x<size; x++)
      fprintf(outfile,"%d\n", picture[y][x]);

  fclose(outfile);
}
```

Listing 1

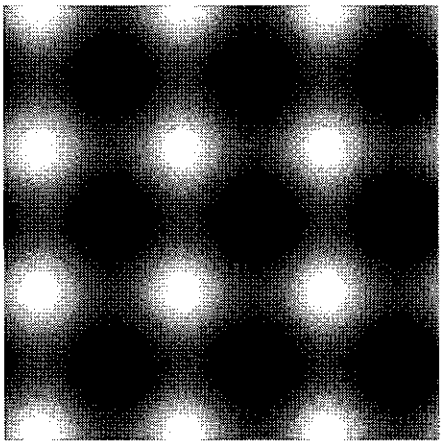

Figure 1

Continued...

Rewrite the program so that it will be in MPI. Only the master should ask the user for input, and only the master should save the output picture. Beware that you must save the output picture in the right order – so be careful how you gather the data back from the slaves.

You must split up the work evenly among all the processors, and you must not hard-code the number of processors.

Make your modifications below – cross out any code you need to cross out, and add any code you need to add. [25 marks]

```
#include <stdio.h>

#include <stdlib.h>

#include <math.h>




int main(int argc, char*argv[])

{ int xmax, ymax, x, y, size;

  double divisor;

  int **picture;

  char buff[1024];

  FILE *outfile;
```

Continued...

```
printf("What file should I store the picture in? ");fflush(stdout);

gets(buff);

printf("How big should the picture be? ");fflush(stdout);

scanf("%d", &size);









divisor = (M_PI*size/60);

picture =(int**)malloc(sizeof(int*)*size);




for (y=0; y<size; y++)

{ picture[y] = (int*)malloc(sizeof(int)*size);

  for (x=0; x<size; x++)

    picture[y][x] = (sin(x/divisor)+cos(y/divisor))*64+128;

}
```

Continued...

```
  outfile=fopen(buff,"w");
  fprintf(outfile,"P2\n%d %d 256\n",size, size);

  for(y=0; y<size; y++)









    for (x=0; x<size; x++)

      fprintf(outfile,"%d\n", picture[y][x]);



  fclose(outfile);

}
```

Continued...

Question 2 [25 marks]

Convert the serial program from *Listing 1* into an OpenMP program. Be careful about the inner loop, and the writing of the output file.

Make your modifications below – cross out any code you need to cross out, and add any code you need to add. [25 marks]

```
#include <stdio.h>

#include <stdlib.h>

#include <math.h>



int main(int argc, char*argv[])

{ int xmax, ymax, x, y, size;

  double divisor;

  int **picture;

  char buff[1024];

  FILE *outfile;



  printf("What file should I store the picture in? ");fflush(stdout);

  gets(buff);

  printf("How big should the picture be? ");fflush(stdout);

  scanf("%d", &size);



  divisor = (M_PI*size/60);

  picture =(int**)malloc(sizeof(int*)*size);
```

Continued...

```
  for (y=0; y<size; y++)
  { picture[y] = (int*)malloc(sizeof(int)*size);
    for (x=0; x<size; x++)
      picture[y][x] = (sin(x/divisor)+cos(y/divisor))*64+128;
  }


  outfile=fopen(buff,"w");
  fprintf(outfile,"P2\n%d %d 256\n",size, size);
  for(y=0; y<size; y++)
    for (x=0; x<size; x++)
      fprintf(outfile,"%d\n", picture[y][x]);









  fclose(outfile);
}
```

Continued...

Question 3 [25 marks]

a) What is the difference between Map and Reduce in Hadoop? [2 marks]

b) What is the difference between Resource Manager, Node Manager, NameNode and Data Node in the Hadoop ecosystem, and how do they relate to the Master, Slave, Yarn and HDFS? [8 marks]

c) If you have a dataset that contains:

- Product Code
- Product Name
- Supplier Code
- Price

Write an algorithm for the map and reduce to calculate the average price for each supplier. [15 marks]

Continued...

Question 4 [25 marks]

Given this serial programme:

```
double complexCalc(double x); // This is a function that
// will take a lot of computing to calculate.

int main(int argc, char *argv[])
{ double total = 0, x;
  int partitions;
  double slice, start, end;

  printf("How many partitions? "); fflush(stdout);
  scanf("%d", &partitions);
  printf(("Start and end point of integration: ");
  fflush(stdout);
  scanf("%f %f"; &start, &end);

  slice = (end-start)/partitions;
  for (x = start + (slice/2); x < end; x = x + slice)
    total = total + f(x);
  total = total * slice;

  printf("The integration is %1.20f\n", total);
}
```

a) When would it be better to convert this into an MPI program instead of an OpenMP, Pthreads, or MapReduce program? Why would this situation be better for MPI and not so good for OpenMP, Pthreads or MapReduce? [10 marks]

Continued...

b) When would it be better to convert this into an OpenMP program instead of an MPI, Pthreads, or MapReduce program? Why would this situation be better for OpenMP and not so good for OpenMP, Pthreads or MapReduce? [10 marks]

c) When is it usually better to use OpenMP instead of MPI? [5 marks]

Continued...

Appendix A: MPI Commands

```
double MPI_Wtime( void )
int MPI_Allgather(const void *sendbuf, int sendcount, MPI_Datatype
  sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, MPI_Comm
  comm)
int MPI_Allreduce(const void *sendbuf, void *recvbuf, int count,
  MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)
int MPI_Barrier( MPI_Comm comm )
int MPI_Bcast( void *buffer, int count, MPI_Datatype datatype, int root,
  MPI_Comm comm )
int MPI_Comm_rank( MPI_Comm comm, int *rank )
int MPI_Comm_size(MPI_Comm comm, int *size)
int MPI_Finalize( void )
int MPI_Gather(const void *sendbuf, int sendcount, MPI_Datatype sendtype,
  void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm
  comm)
int MPI_Init( int *argc, char ***argv )
int MPI_Recv(void *buf, int count, MPI_Datatype datatype, int source, int
  tag, MPI_Comm comm, MPI_Status *status)
int MPI_Reduce(const void *sendbuf, void *recvbuf, int count, MPI_Datatype
  datatype, MPI_Op op, int root, MPI_Comm comm)
int MPI_Scatter(const void *sendbuf, int sendcount, MPI_Datatype sendtype,
  void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm
  comm)
int MPI_Send(const void *buf, int count, MPI_Datatype datatype, int dest,
  int tag, MPI_Comm comm)
int MPI_Wait(MPI_Request *request, MPI_Status *status)
```

Appendix B: OpenMP Commands

#pragma omp

- copyin
- default
- firstprivate
- for
- lastprivate
- num_thrads
- parallel
- private
- reduction
- schedule
- shared

End of Paper